Editorial Torneo de Lanzamiento de Objetos

José Santiago Vales Mena

Junio 16, 2024

Subtarea 1 [7 puntos]

La primera observación que debemos hacer es que todas las operaciones las podemos hacer con enteros. Si $\frac{a}{b} \geq D$, entonces la división entera de $\frac{a}{b}$ es al menos D. (Usa `long long` porque la multiplicación se puede desbordar).

Cuando solo tenemos un objeto y una persona, basta verificar si ese objeto y esta persona pasan el lanzamiento.

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    int N, M, D;
    cin >> N >> M >> D;
    int64_t f, a, p;
    cin >> f >> a >> p;

    if (f * a / p >= D) cout << 1;
    else cout << 0;

    return 0;
}
```

Subtarea 2 y 3 [16 puntos]

Cuando solo hay una persona o un objeto, basta probar si la persona (objeto) lanzando todos los objetos (lanzado por todas las personas) es al menos D.

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    int N, M, D;
    cin >> N >> M >> D;
    int64_t f, a;
    cin >> f >> a;

    int ans = 0;
    for (int i = 0; i < M; i++)  {
        cin >> p;
        if (f * a / p >= D) ans++;
    }

    cout << ans;
```

```
    return 0;
}
```

Subtarea 4 [35 puntos]

Para esta subtarea, corremos todas las personas contra todos los objetos y contamos la cantidad de puntos. Esto tiene una complejidad de $O(N \times M)$, por lo que no obtiene todos los puntos, ya que en el peor de los casos es $O(10^{10})$.

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    int N, M, D;
    cin >> N >> M >> D;
    vector<int64_t> f(N), a(N), p(M);

    for (int i = 0; i < N; i++) {
        cin >> f[i] >> a[i];
    }
    for (int i = 0; i < M; i++) {
        cin >> p[i];
    }

    int ans = 0;
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < M; j++) {
            if (f[i] * a[i] / p[j] >= D) ans++;
        }
    }

    cout << ans;
    return 0;
}
```

Subtarea 5 [15 puntos]

Si todos los objetos pesan 1, entonces, únicamente basta comprobar si cada persona puede lanzar todos los objetos. Esto se traduce a verificar que $f_i \times a_i \geq D$. Si es así, a nuestro contador de respuesta le sumamos M, indicando que esa persona puede lanzar todos los objetos. Recuerda usar `long long` en tu respuesta porque en el peor de los casos todas las combinaciones son posibles $N \times M$.

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    int N, M, D;
    cin >> N >> M >> D;
    int64_t f, a;
    int64_t ans = 0;

    for (int i = 0; i < N; i++) {
        cin >> f >> a;
        if (f * a >= D) ans += M;
    }
```

```
    cout << ans;
    return 0;
}
```

Subtarea 6 [100 puntos]

Solución 1 (Búsqueda binaria)

Digamos que queremos ver cuántos objetos puede lanzar la persona con estadísticas $(f_i, a_i) $. Sea p_1 y p_2 objetos diferentes. Es claro que,

$$p_1 \leq p_2 \implies \frac{f_i \times a_i}{p_1} \geq \frac{f_i \times a_i}{p_2}.$$

En palabras del problema, esto quiere decir que entre más pesado es un objeto, más difícil es que supere la distancia D. Ordenemos los objetos por peso, donde p_1 es el más ligero y p_M el más pesado. Para cierta persona, nuestra tarea es encontrar el objeto más pesado que puede lanzar y que supere la distancia D. Si sabemos que p_k es el objeto más pesado que puede lanzar y que supere la distancia, entonces esa persona puede lanzar k objetos que superen D. Para hacer esa búsqueda, podemos ayudarnos de la búsqueda binaria y de esta forma nuestra complejidad se vuelve $O(N \log M)$.

```
#include <bits/stdc++.h>
using namespace std;

int N, M, D;
vector<int64_t> personas;
vector<int64_t> pesos;

int binaria(int ini, int fin, int64_t si) {
    if (ini == fin) return ini;
    int mit = (ini + fin) / 2;
    if (si / pesos[mit] >= D) return binaria(mit + 1, fin, si);
    else return binaria(ini, mit, si);
}

int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cin >> N >> M >> D;

    personas.resize(N);
    pesos.resize(M);

    for (int i = 0; i < N; i++) {
        cin >> personas[i];
        int64_t x; cin >> x;
        personas[i] *= x;
    }

    for (int i = 0; i < M; i++) cin >> pesos[i];
    sort(pesos.begin(), pesos.end());
    sort(personas.begin(), personas.end());

    int64_t ans = 0;
    for (int i = 0; i < N; i++) {
        ans += binaria(0, M, personas[i]);
    }
```

```
    cout << ans;

    return 0;
}
```

Solución 2 (Dos punteros)

Usando el análisis de la solución anterior, podemos asignar a cada persona un puntaje físico $s_i = a_i \times f_i$. Ahora ordenamos igualmente a las personas por ese puntaje. Imagina que la persona s_0 puede lanzar hasta el objeto p_k. ¿Qué información útil nos da para la persona s_1? Pues si tiene un puntaje físico mayor, sabemos que todos los que el anterior pudo lanzar, esta persona también podrá lanzarlos. Por lo tanto, podemos usar una técnica de dos punteros donde usamos la información de la persona anterior para continuar nuestra búsqueda. Esto nos deja con una complejidad de $O(N + M)$, que resulta más eficiente que la binaria.

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    int N, M, D;
    cin >> N >> M >> D;
    vector<int64_t> personas(N);
    vector<int64_t> pesos(M);
    for (int i = 0; i < N; i++) {
        cin >> personas[i];
        int64_t x; cin >> x;
        personas[i] *= x;
    }

    for (int i = 0; i < M; i++) cin >> pesos[i];
    sort(pesos.begin(), pesos.end());
    sort(personas.begin(), personas.end());

    int64_t ans = 0;
    for (int i = 0, j = 0; i < N; i++) {
        while (j < M && personas[i] / pesos[j] >= D) j++;
        ans += j;
    }

    cout << ans;

    return 0;
}
```